High Performance Computing by scaling Linux

Linux Symposium Japan
High Performance Computing
Linux Memory Management

2007-03-13

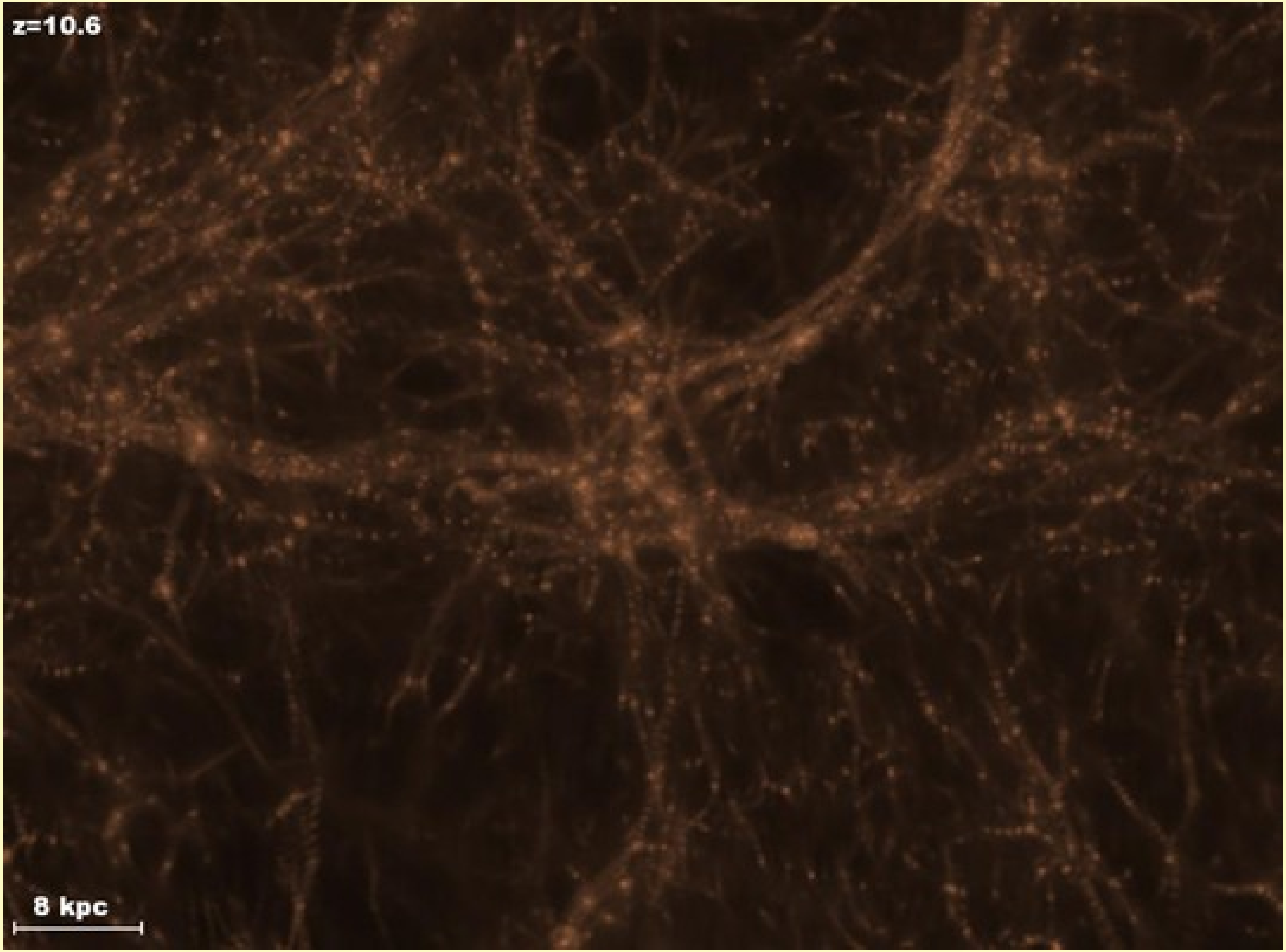
Christoph Lameter, Ph.D.
christoph@lameter.com
SGI
Linux Foundation

Overview

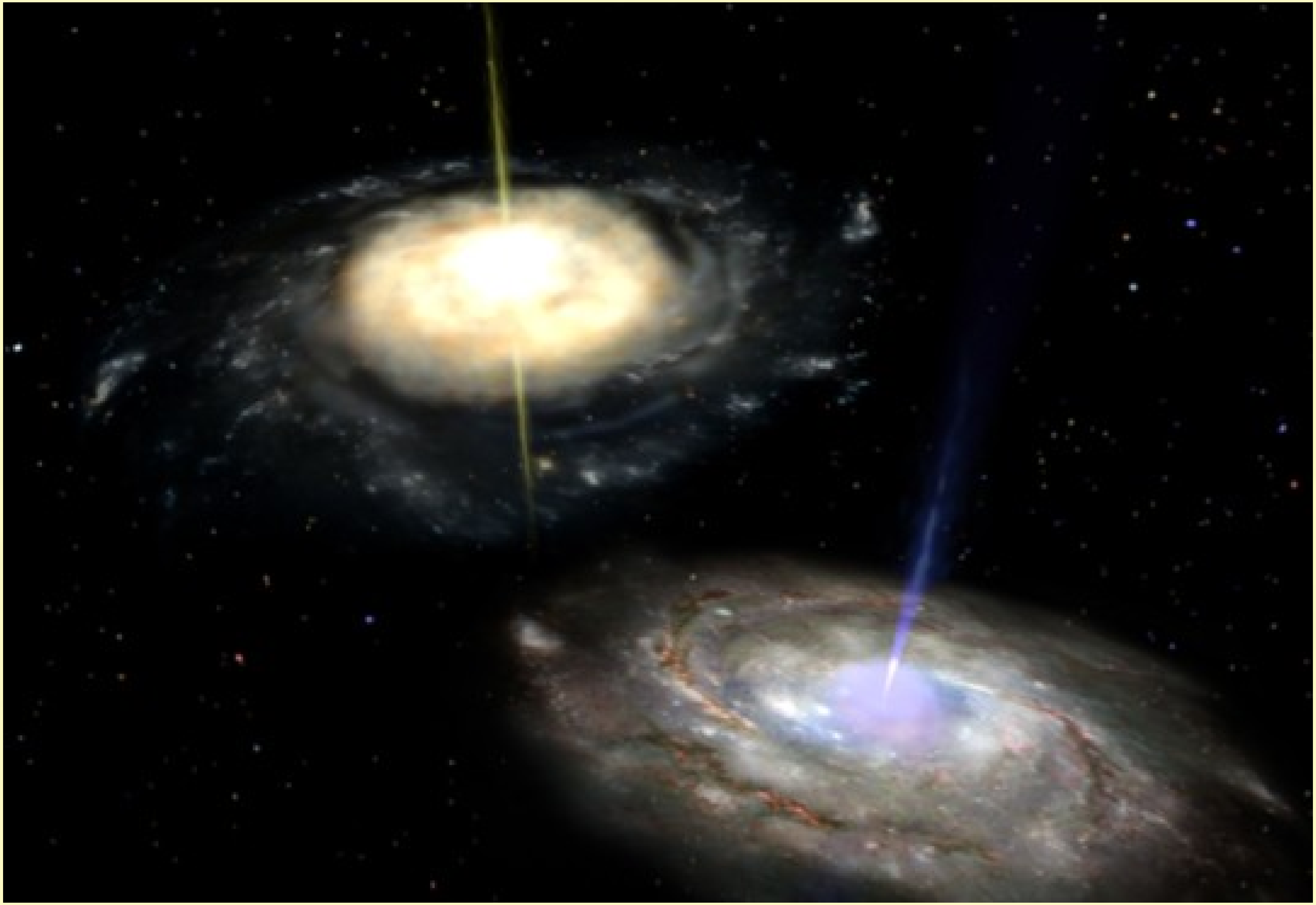
- Some example of the use of High Performance Computing
- Computer Architectures in use for HPC Computing
  - Cluster
  - Supercomputer
  - Mainframe
- Scaling Linux on a Supercomputer.
- Memory management
  - NUMA
  - VM tuning
- Memory Control
  - Application allocation control
  - Migrating Memory
- Performance numbers

NASA Columbia Supercomputer with 10240 processors

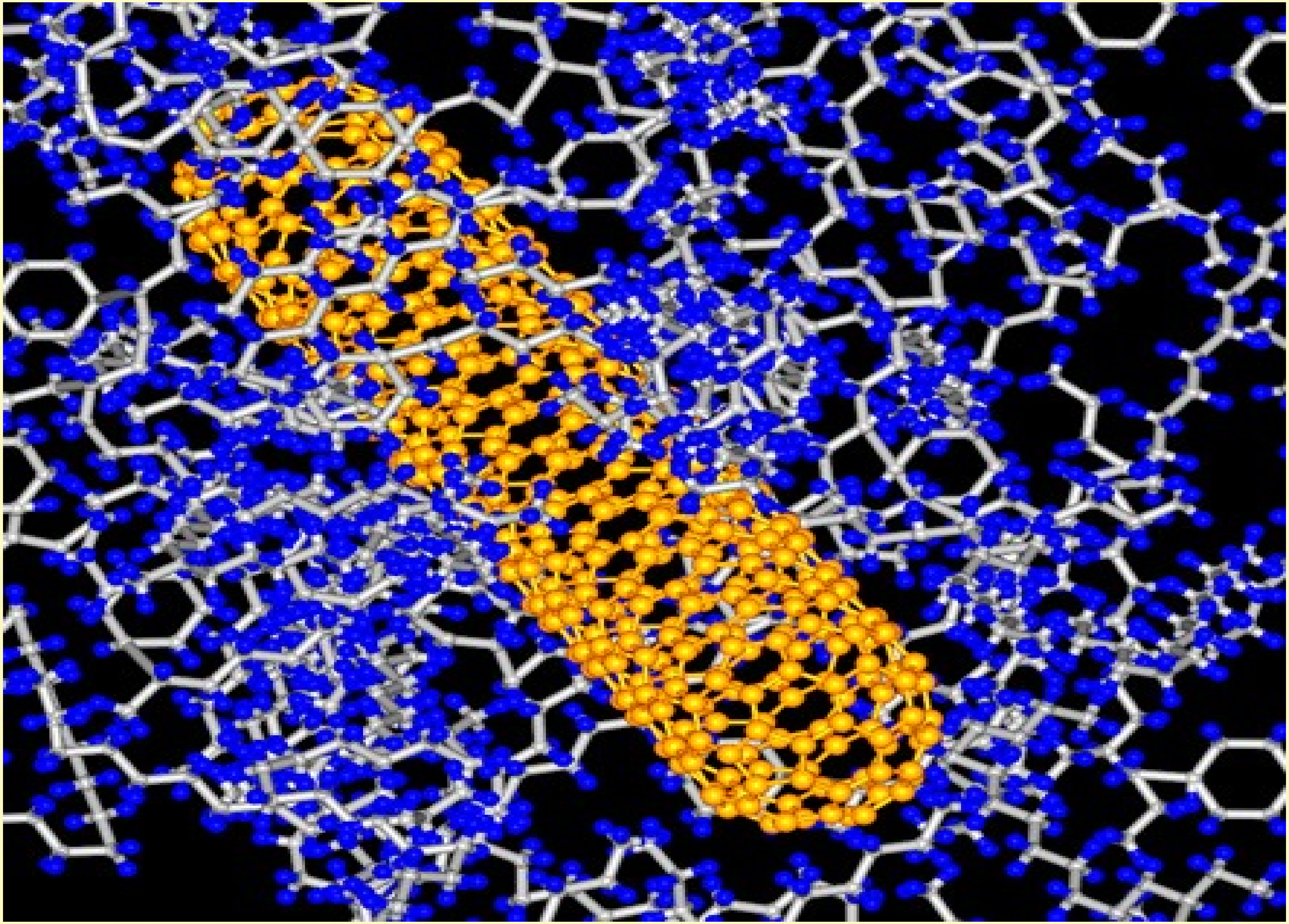
Dark Matter Halo Simulation for the Milky Way



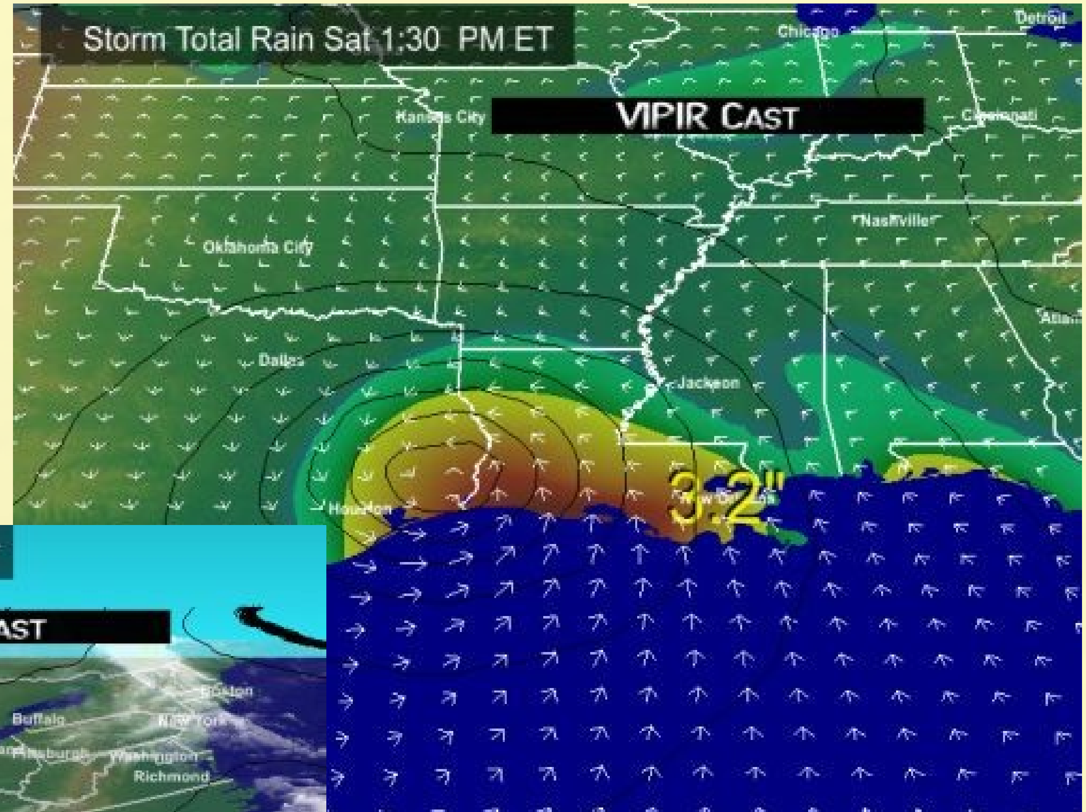
Black Hole Simulation



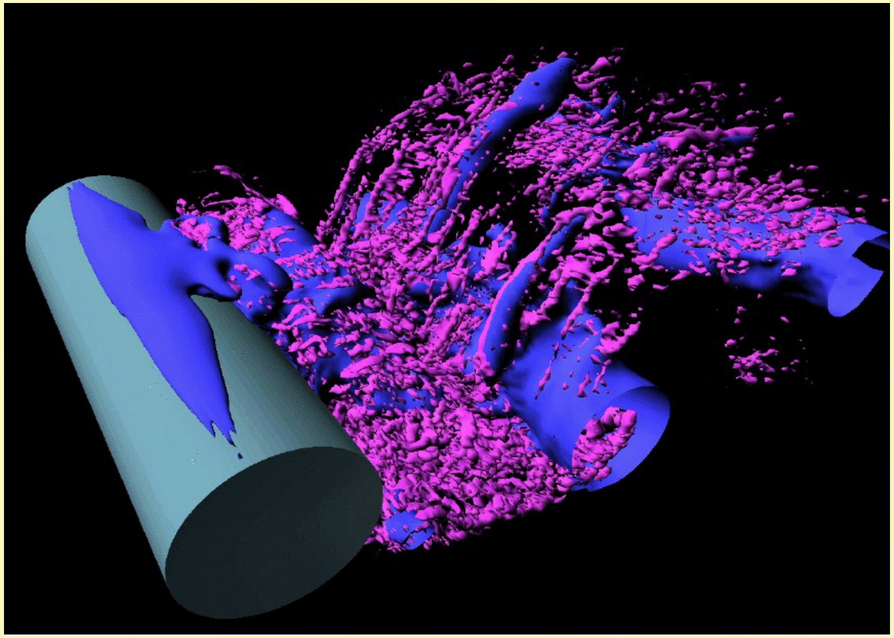
Carbon Nanotube-polymer composite material

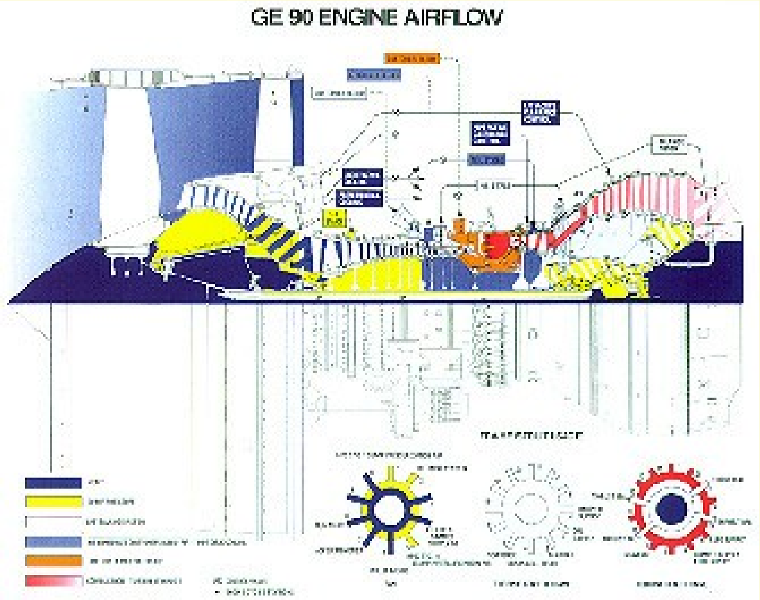


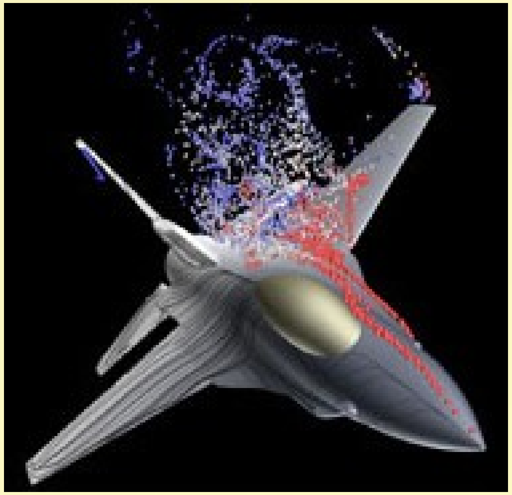
Forecast of Hurrican Katrina



Airflow Simulations







Applications of High Performance Computing

- Solve complex computationally expensive problems
- Scientific Research
  - Physics (quantum mechanics, nuclear phenomena)
  - Cosmology
  - Space
  - Biology (gene analysis, virus, bacteria etc)
- Simulations
  - Weather (Hurricanes)
  - Study of molecules and new substances
- Complex data analysis
- 3D design
  - Interactive modeling (f.e. car design, aircraft design)
  - Structural analysis.

High Performance Computer Architectures

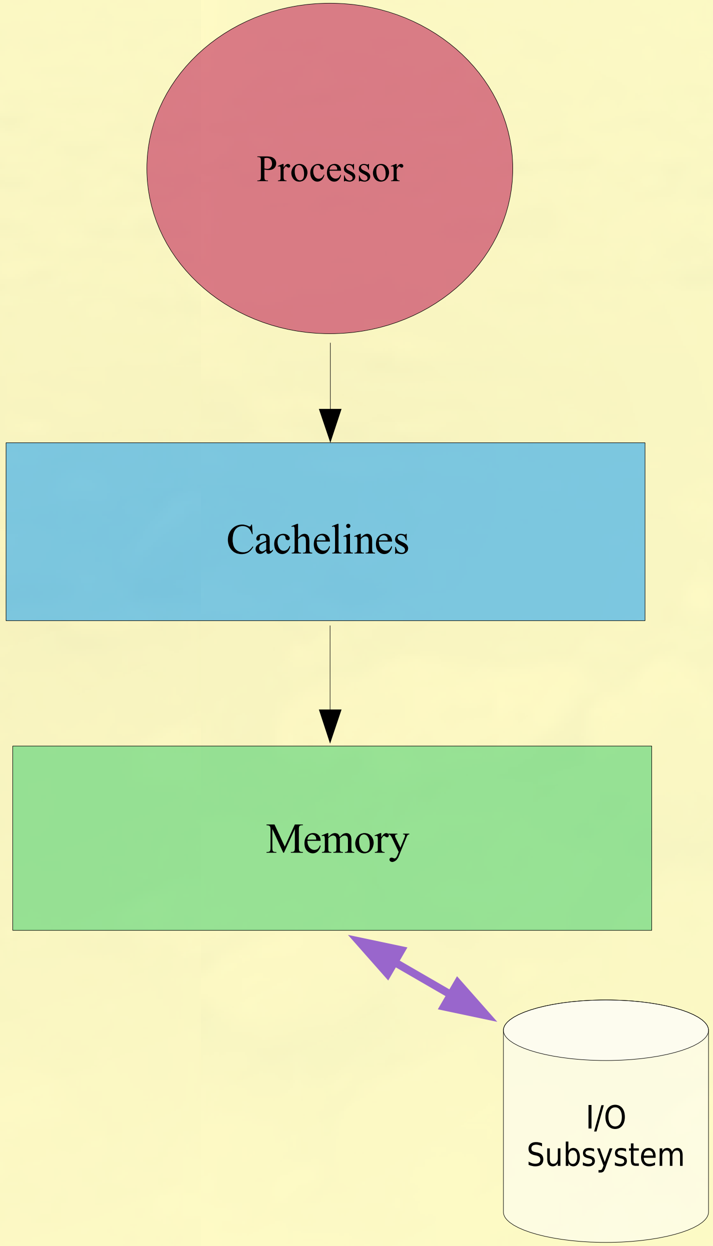
- Supercomputer
  - Single memory space
  - NUMA architecture. Memory nodes / Distant memory.
  - Challenge to scale the Operating System
- Cluster
  - Multiple memory spaces
  - Networked commodity servers
  - Network communication critical for performance
  - Challenge to redesign applications for a cluster
- Mainframe
  - Singe uniform memory space with multiple processors
  - Scalable I/O subsystem
  - Mainly targed to I/O transactions
  - Reliable and maintainable (24 by 7 availability)

Linux on Supercomputers

- Operating System enhancements are needed for Supercomputers support.
- Processors
  - 8 processor to thousands (10K @NASA) at the high end.
- Memory
  - 32GB to 16TB (plans exist to support at least 1 Peta byte of main memory in the near term future)
- Physical size
  - Large Rack to customized buildings (f.e. LRZ Munich, NASA, APAC Australia)
- I/O
  - Large Storage farms with hundreds of petabytes of hard disk store
  - Robotic systems to access archives of tapes for long term storage

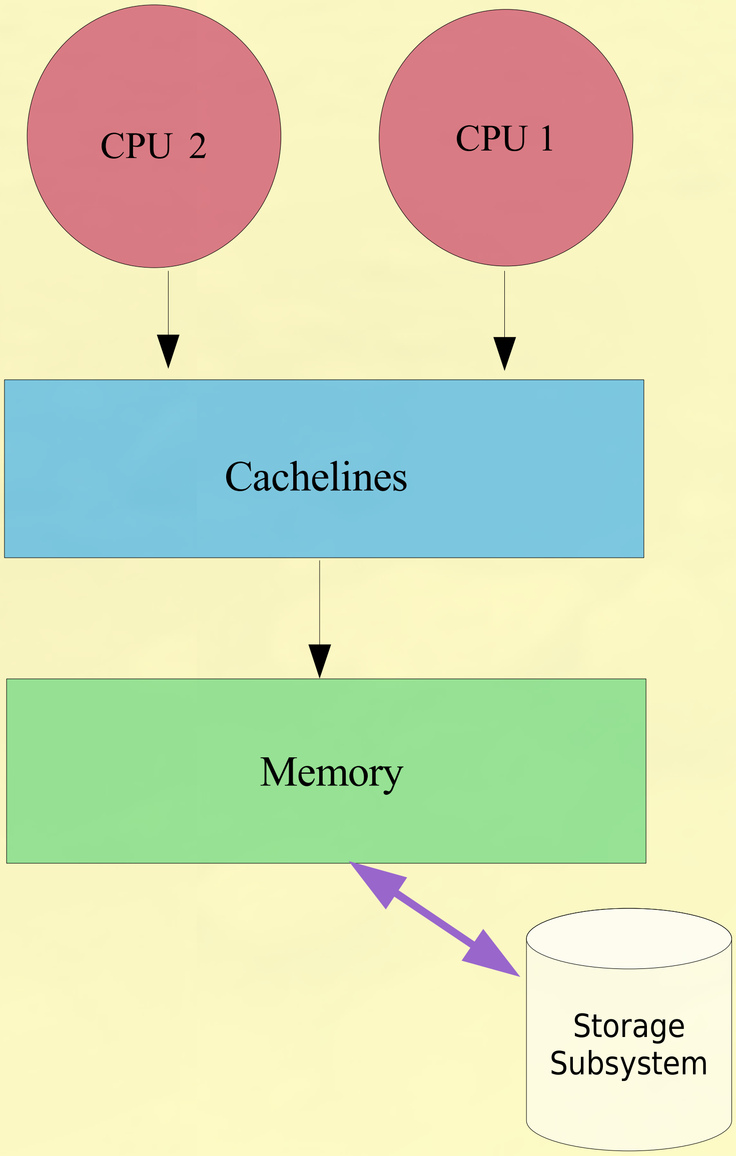
Single Processor System

- All computation on a single processor
- Only parallelism that needs to be managed is with the I/O subsystem
- Memory is slow compared to the processor.
- Speed of the system depends on the effectiveness of the cache
- Memory accesses have the same performance.



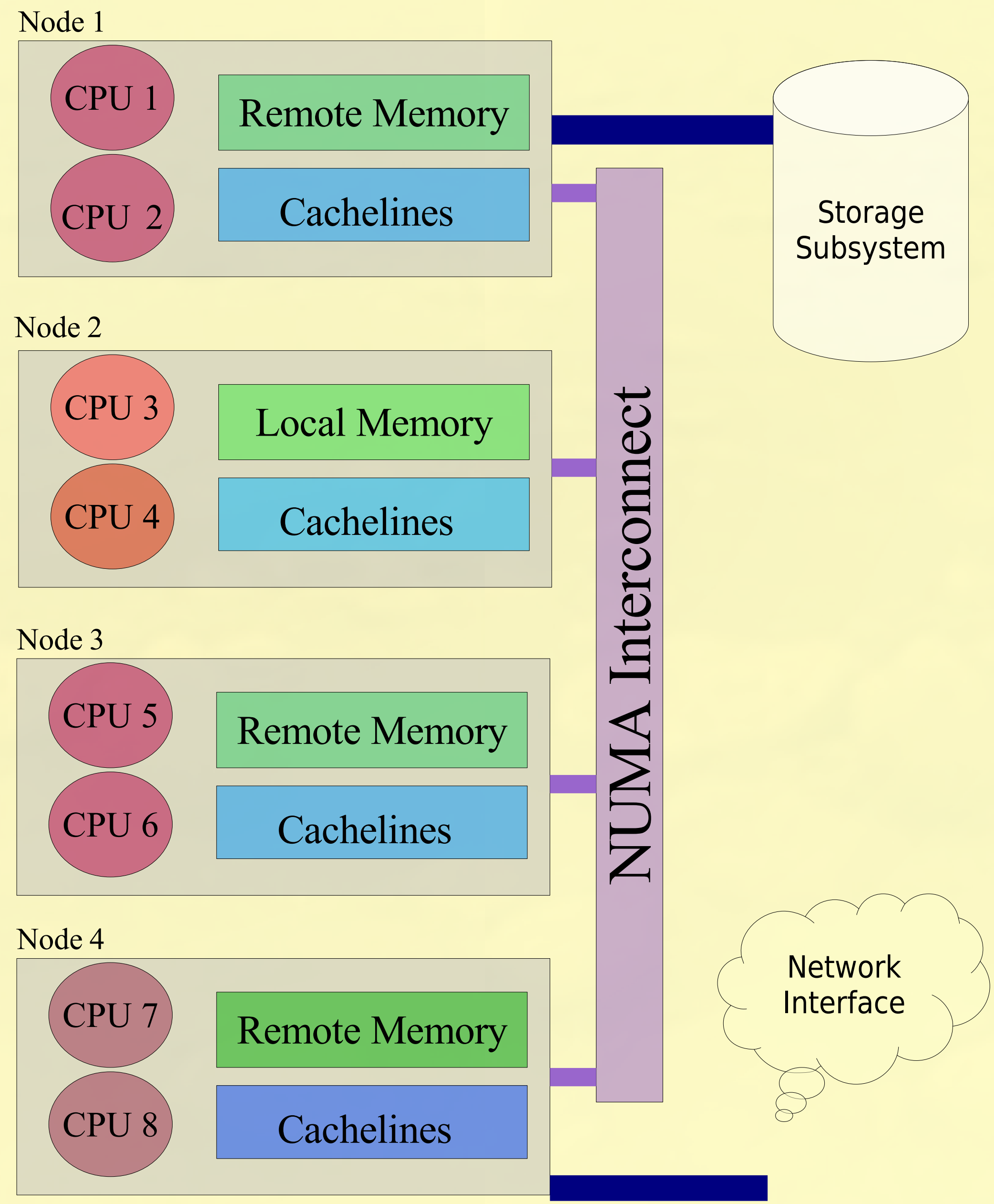
Symmetric Multi Processing (SMP)

- Multiple processors
- New need for synchronization between processors
- Cache control issues
- Performance enhancement through multiple processors working independently
- Cacheline contention
- Data layout challenges: shared vs. processor local
- All memory access have the same performance



Non Uniform Memory Architecture (NUMA)

- Multiple SMP like systems called “nodes”
- Memory at various distances (NUMA)
- Interconnect
- MESI type cache coherency protocols
- SLIT tables
- Memory Placement
- Node Local from node 2 processor 3
- Device Local

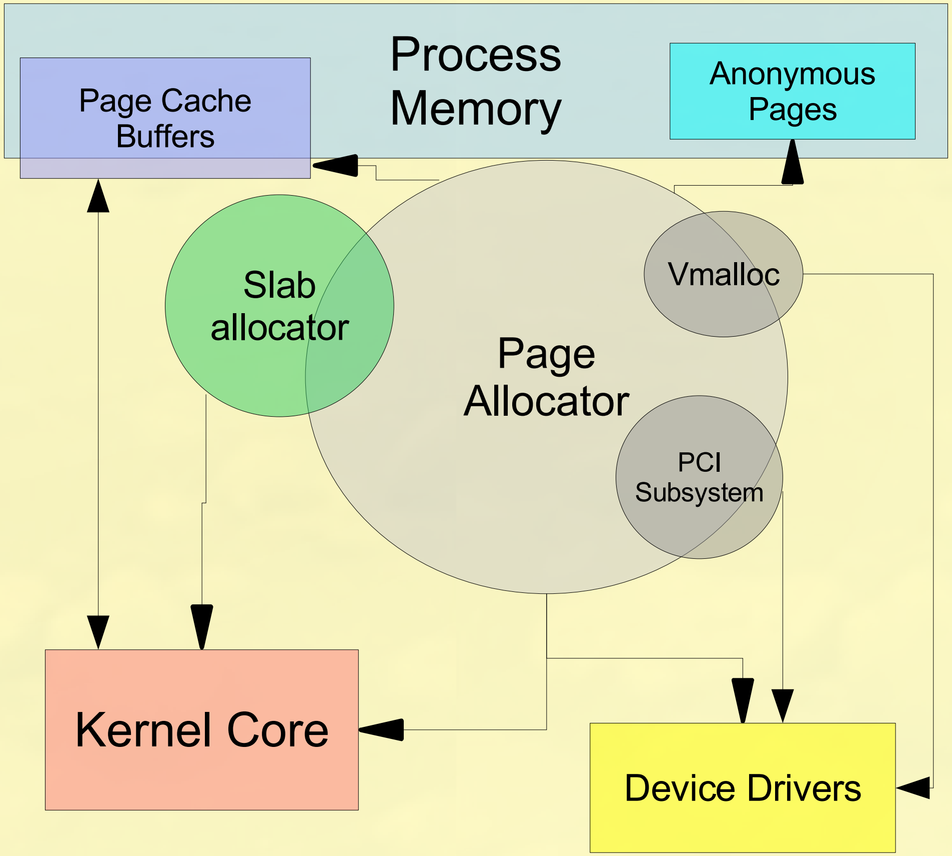


Scaling up Linux

- Per cpu areas
- Per node structures
- Memory allocators aware of distance to memory
- Lock splitting
- Cache line optimization
- Memory allocation control from user space
- Sharing is a problem
- Local Memory is the best
- Larger distances mean larger systems are possible
- The bigger the system the smaller the portion of local memory.

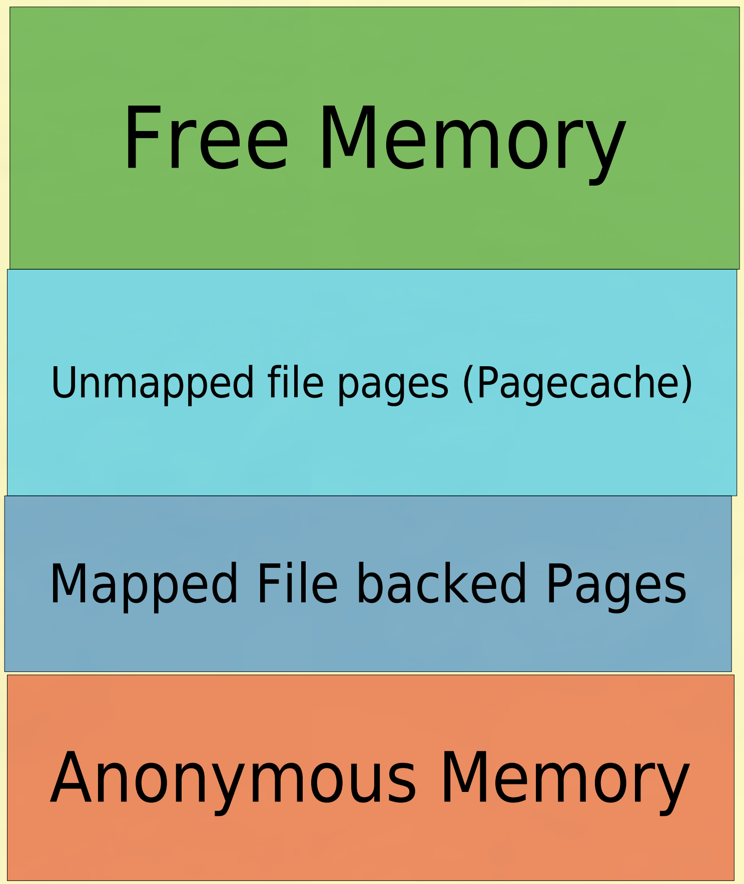
Allocators for a Uniform Memory Architecture

- Page Chunks
- Page allocator
- Anonymous memory
- File backed memory
- Swapping
- Slab allocator
- Device DMA allocator
- Page Cache
- read() / write()
- Mmapped I/O.



UMA Memory Reclaim

- Anonymous memory freed when a process terminates
- Mapped file backed pages become unmapped but are not freed. So unmapped file pages accumulate.
- If memory runs low the swapper begins reclaim of memory.
- Light reclaim just frees unmapped file pages.
- If memory stays tight then memory may be unmapped which will allow the freeing of mapped file backed pages and the swapping out of anonymous pages.

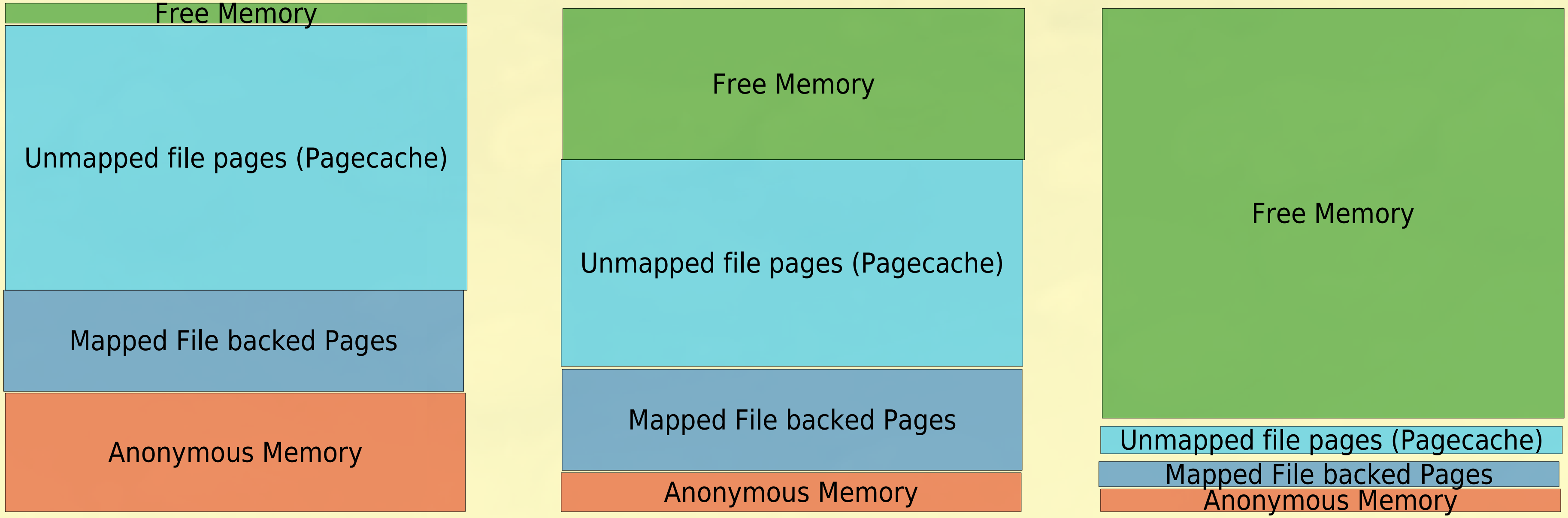


NUMA Allocators

- Memory management per node
- Memory state and possibilities of allocation
- Traversal of the zonelist (or nodelist)
- Process location vs. memory allocation
- Scheduler interactions
- Predicting memory use?
- Memory load balancing
- Support to shift the memory load

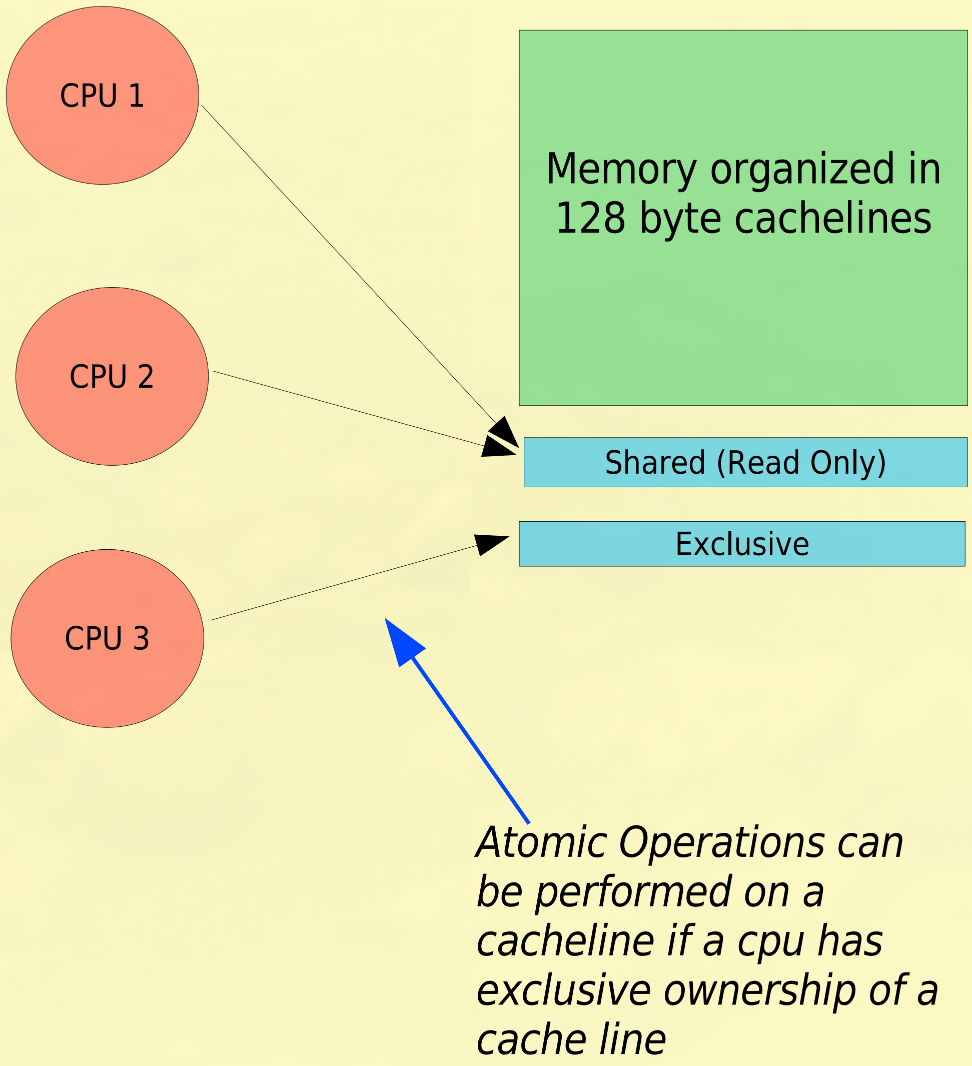
NUMA standard reclaim

- Reclaim is a global reclaim.
- Reclaim only active if total free memory becomes low
- No local reclaim results in lots of node allocations
- Off node allocations occur until all the nodes are out of memory. On a large NUMA system this can seem to never occur.
- The overhead of remote memory access becomes excessively large.
- One technique used in the past is to manually drop the pagecache.

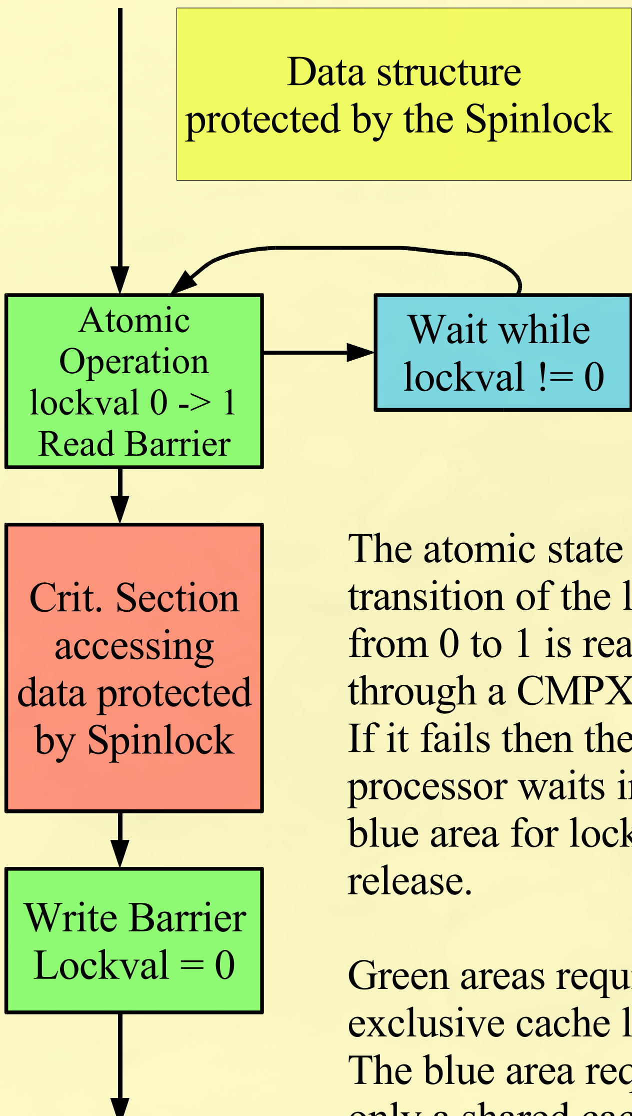


Cache lines influence Performance

- Modes of Cache lines
  - Shared
  - Exclusive
- Cache Lines
  - Efficiency
  - Optimization
  - Bouncing
- Special Operations
  - Read Modify Write



Spinlock Behavior



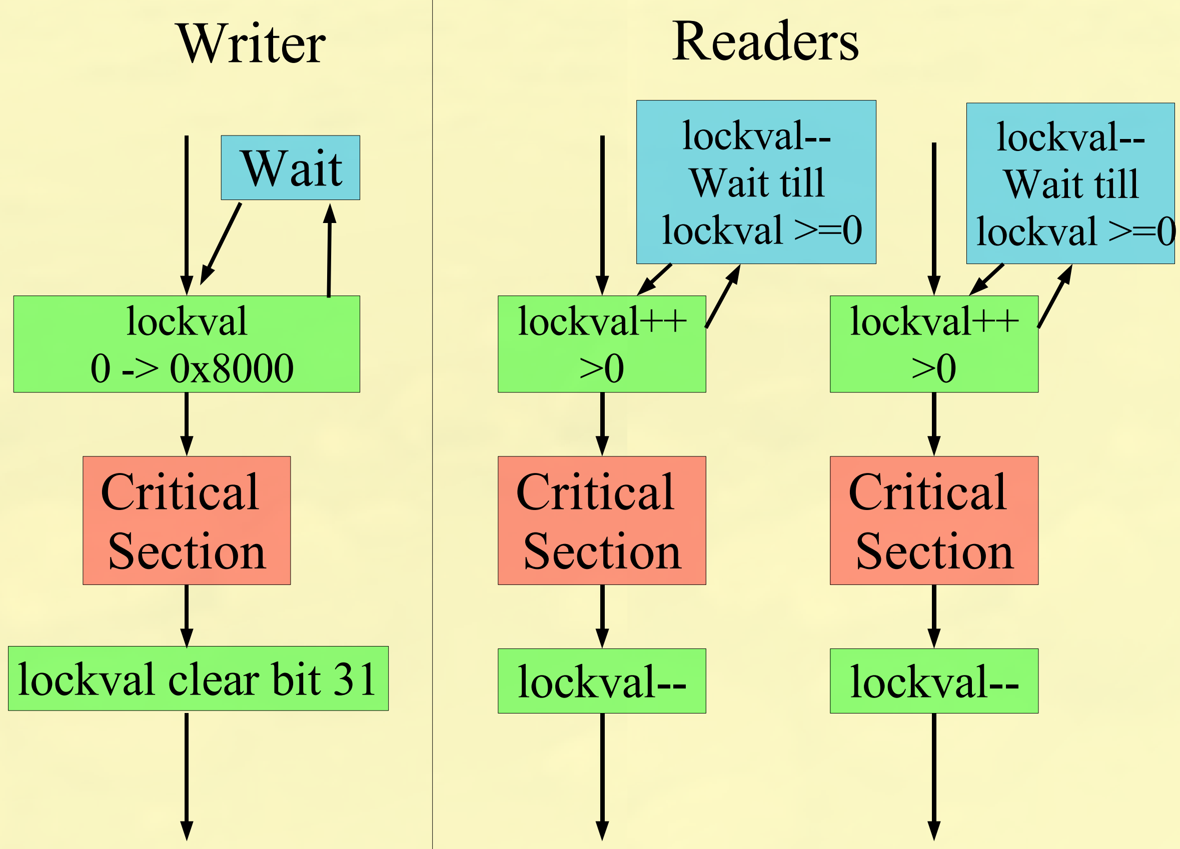
The atomic state transition of the lockval from 0 to 1 is realized through a CMPXCHG. If it fails then the processor waits in the blue area for lock release.

Green areas require exclusive cache lines The blue area requires only a shared cache line.

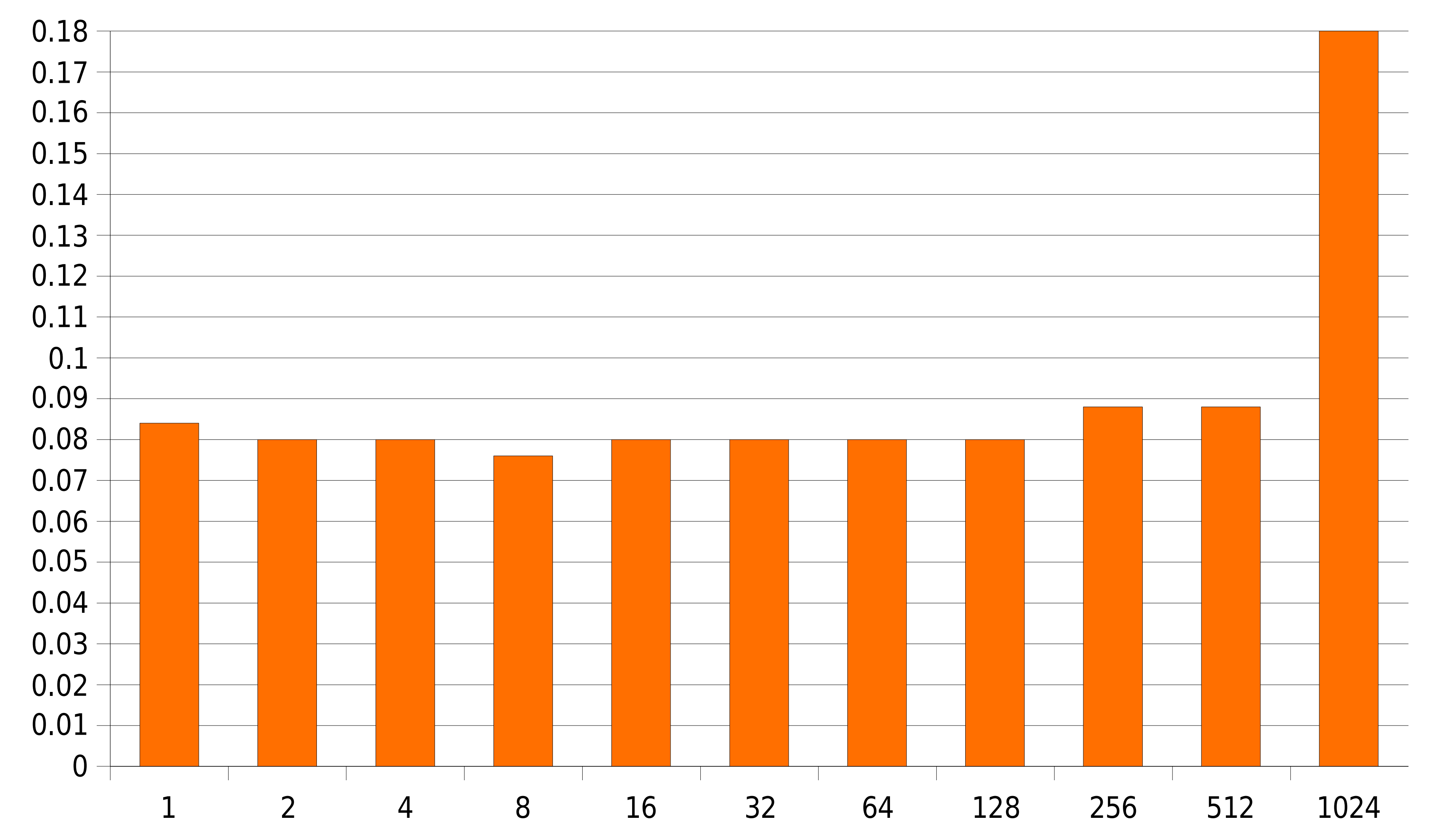
- Protected Data
- Critical Sections
- Locking
- Unlocking
- Exclusive Cache line use vs. Shared Cache line
- Bouncing Cachelines
- Spinlocks under contention

Reader/Writer Spinlocks

- Lock value
  - >0 -> nr readers
  - <0 writer
  - 0 free
- Needs
  - 2x Cmpxchg
  - 2x Fetchadd
  - Clear Bit 31 (byte store instead?)
- Performance worse than regular spinlock



Parallel processes with independent memory spaces allocating 100 Megabytes concurrently



Threads in a single memory space allocating 100 Megabytes concurrently